

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**



## PATENT ABSTRACTS OF JAPAN

(11) Publication number: **08161169 A**

(43) Date of publication of application: **21.06.96**

(51) Int. Cl. **G06F 9/38**  
**G06F 9/28**  
**G06F 9/45**

(21) Application number: 06305907

(22) Date of filing: 09.12.94

(71) Applicant: **TOSHIBA CORP**

(72) Inventor: TAKEUCHI YOICHIRO

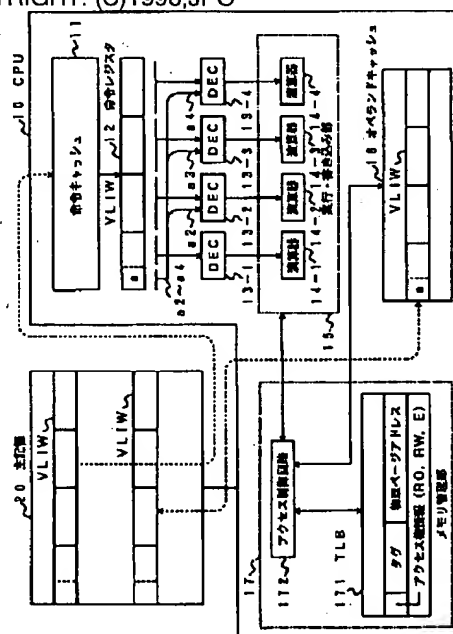
## (54) VLIW TYPE COMPUTER SYSTEM AND METHOD FOR INTERPRETING/ EXECUTING VLIW

COPYRIGHT: (C)1996.JPO

(57) Abstract:

**PURPOSE:** To reduce a useless storage area due to a code part conventionally occupied by a nop instruction on an VLIW (very long instruction word).

**CONSTITUTION:** Instruction validating information (a) consisting of bits a2 to a4 indicating which instruction field (invalid instruction field) does not require interpretation/execution out of a 2nd instruction field and after is formed on a part of the leading field of a VLIW having four instruction fields, and when the VLIW is fetched in an instruction register 12, the contents (excluding the information (a)) of the 1st instruction field are unconditionally interpreted by a decoder 13-1 to control a computing element 14-1 and the contents of the 2nd to 4th instruction fields are interpreted by respective decoders 13-2 to 13-4 in accordance with the states of the bits a2 to a4 in the information (a) to control computing elements 14-2 to 14-4, so that the invalid instruction field part can be utilized as a variable/constant area.



52# (✓)  
①

(19) 日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平 8 - 1 6 1 1 6 9

(43) 公開日 平成 8 年 (1996) 6 月 21 日

(51) Int. Cl. °	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F	9/38	3 7 0 X		
	9/28	3 2 0	7230 - 5 B	
	9/45			
		7737 - 5 B	G 0 6 F	9/44
		7737 - 5 B		3 2 2 E
				3 2 2 J
審査請求	未請求	請求項の数 6	O L	(全 2 3 頁)

(21) 出願番号 特願平 6 - 305907

(22) 出願日 平成 6 年 (1994) 12 月 9 日

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町 72 番地

(72) 発明者 竹内 陽一郎

東京都府中市東芝町 1 番地 株式会社東芝

府中工場内

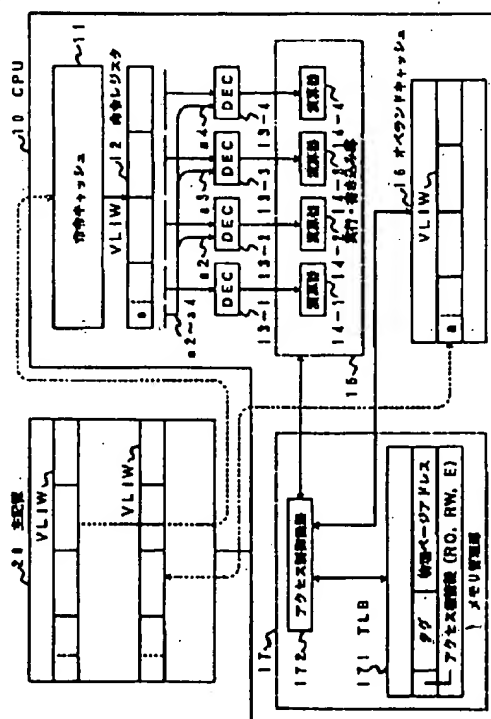
(74) 代理人 弁理士 鈴江 武彦

(54) 【発明の名称】 V L I W 方式の計算機システム及び V L I W の解釈・実行方法

(57) 【要約】

【目的】従来 V L I W 上の n o p 命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができるようにする。

【構成】4 つの命令フィールドを持つ V L I W の先頭フィールドの一部に、2 番目以降の命令フィールドのうち、解釈・実行の必要のない命令フィールド（無効命令フィールド）がどれであることを示すビット a 2 ~ a 4 からなる命令有効化情報 a を設け、当該 V L I W が命令レジスタ 1 2 にフェッチされた際には、1 番目の命令フィールドの（命令有効化情報 a を除く）内容を無条件でデコーダ 1 3 - 1 により解釈して演算器 1 4 - 1 を制御すると共に、2 番目 ~ 4 番目の命令フィールドの内容を命令有効化情報 a 中のビット a 2 ~ a 4 の状態に応じてデコーダ 1 3 - 2 ~ 1 3 - 4 により解釈して演算器 1 4 - 2 ~ 1 4 - 4 を制御することで、無効命令フィールド部分を変数/定数領域として利用可能な構成とする。



## 【特許請求の範囲】

【請求項1】 並列動作可能な複数の演算器と、これら各演算器を個々に制御するための同数の独立した命令フィールドを持つ1個の長い命令語VLIWであって、その一部に解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報を有し、次に解釈・実行すべき命令語VLIWを保持するための命令レジスタと、

この命令レジスタに保持されている前記命令語VLIWの各命令フィールドに対応してそれぞれ設けられ、対応する命令フィールドの内容を解釈して当該フィールドに対応する前記演算器を制御するデコード手段であって、当該フィールドが前記命令有効化情報により無効命令フィールドであることが示されている場合には、当該フィールドの内容に従う対応する前記演算器の動作を禁止するデコード手段と、

前記命令有効化情報により無効であることが示されている前記命令語VLIW上の命令フィールドに対して、演算オペランドとしてアクセスすることを許すメモリ管理手段とを具備することを特徴とするVLIW方式の計算機システム。

【請求項2】 ソースプログラムをコンパイルして前記計算機システム内で実行される命令語VLIWの列からなるオブジェクトプログラムを生成するコンパイル手段であって、前記命令語VLIW中の解釈・実行が不要な無効命令フィールドを演算オペランド領域として、当該フィールドに定数または変数を割り当てるコンパイル手段を更に具備することを特徴とする請求項1記載のVLIW方式の計算機システム。

【請求項3】 並列動作可能な複数の演算器を同数の独立した命令フィールドを持つ1個の長い命令語VLIWにより制御するVLIW方式の計算機システムにおいて、

前記命令語VLIWの列からなるオブジェクト群をリンクしてロードモジュールを生成するリンク手段であって、当該ロードモジュールの生成時に、前記演算器を動作させる有効命令フィールドを除く無効命令フィールド部分を削除するリンク手段と、

前記ロードモジュールから指定ページの命令語VLIWの列を主記憶上にロードするためのデマンドページアクセス要求が発生した場合に、前記無効命令フィールド部分を含む命令語VLIWの列を復元するデマンドページアクセス手段とを具備することを特徴とするVLIW方式の計算機システム。

【請求項4】 前記命令語VLIWは、その一部に解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報を有しており、前記リンク手段は、前記各命令語VLIW毎に、そのVLIWの命令有効化情報をもとに前記無効命令フィールドの位置を確認して、当該フィールドを取り除き、

前記デマンドページアクセス手段は、前記ロードモジュールから指定ページの命令語VLIWの列を主記憶上にロードする際に、対応する前記命令語VLIWの命令有効化情報を参照して、当該VLIW上の前記無効命令フィールドの位置を検出し、その位置を飛ばして当該VLIW上の有効命令フィールドの内容を前記主記憶に配置することを特徴とする請求項3記載のVLIW方式の計算機システム。

【請求項5】 前記リンク手段は、前記リンク時に、リンク前の前記命令語VLIWの列における前記有効命令フィールドと無効命令フィールドの配置を示す命令マップを生成して前記ロードモジュールに付加し、前記デマンドページアクセス手段は、前記ロードモジュールから指定ページの命令語VLIWの列を主記憶上にロードする際に、当該ロードモジュールに付されている前記命令マップに従って、前記無効命令フィールドの位置を検出し、その位置にノーオペレーション命令nopを配置することを特徴とする請求項3記載のVLIW方式の計算機システム。

【請求項6】 並列動作可能な複数の演算器を同数の独立した命令フィールドを持つ1個の長い命令語VLIWにより制御するVLIW方式の計算機システムに適用されるVLIWの解釈・実行方法において、前記命令語VLIWの一部に、解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報を設け、解釈・実行すべき前記命令語VLIW上に、当該VLIWの前記命令有効化情報により無効であることが示されている命令フィールドが存在する場合には、その命令フィールドの内容に従う対応する前記演算器の動作を禁止し、

前記命令有効化情報により無効であることが示されている前記VLIW上の命令フィールドに対して、演算オペランドとしてアクセスすることを許すことにより、当該フィールドの演算オペランド領域としての利用を許すようにしたことを特徴とするVLIWの解釈・実行方法。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】本発明は、並列動作可能な複数の演算器を独立した複数の命令（命令フィールド）の集まりであるVLIW (Very Long Instruction Word) と称される1個の長い命令語により制御するVLIW方式の計算機システム及びVLIWの解釈・実行方法に関する。

## 【0002】

【従来の技術】近年、処理の高速化のために、VLIW方式の計算機システムが開発されている。この種の計算機システムは、並列動作可能な複数の演算器を備えており、VLIW上の独立した各命令により、当該各演算器

を同期をとって個々に且つ並列に制御するようになって  
いる。

【0003】図13は、VLIW方式の計算機システムの  
要部の基本構成を示すものである。この図13のシス  
テムでは、命令#1～#4の4命令を1組とするVL  
IW131により、4つの演算器132-1～132-4が制  
御される構成となっている。即ち、図13のシステムに  
おいて、VLIW131中の各命令#1～#4は、それ  
ぞれデコード(DEC)133-1～133-4により同時  
に且つ並列に解釈され、それぞれ対応する演算器132  
-1～132-4の動作を制御する。

【0004】このように、VLIW方式の計算機シス  
テムでは、VLIWにより複数の演算器を並列動作させ  
ることができる。したがって、プログラマまたはコンパイ  
ラが、並列に実行可能な演算を調べ、これらの実行をV  
LIW上の対応する命令にプログラミングすることで、  
命令語レベルの並列処理が実現され、処理の高速化が図  
れる。

【0005】さて、この種の計算機システムで使用され  
るVLIW方式のプログラム(オブジェクトプログラ  
ム)では、実行する計算アルゴリズムの因果関係(各命  
令間の依存関係)によっては、本来並列動作可能な複数  
の演算器の全てを演算に使用することができないケース  
が発生する。

【0006】このような場合、使用しない演算器に対応  
するVLIW中の命令フィールドには、演算を行わない  
ことを明示するノーオペレーション命令(nop命令)  
が配置される。このnop命令の配置により、誤動作防  
止が図られる。

【0007】

【発明が解決しようとする課題】上記したように、VL  
IW方式の計算機システムでは、各命令間の依存関係に  
より同時に実行できる命令が所定数揃わない場合、VL  
IW中の一部の命令フィールドに誤動作防止のためにn  
op命令が配置されるようになっていた。

【0008】このため、演算器の数を増やし、それに合  
わせてVLIWの命令語長を極端に長くした場合や、も  
ともと並列動作しにくい処理をプログラミングした場合  
などには、実行コードの大半がnop命令となってい  
てしまふ。オブジェクトプログラムサイズ(実行オブジェ  
クトサイズ)が著しく大きくなって、そのプログラムの占め  
る主記憶領域も大きくなるという問題があった。このこ  
とは、ディスク(補助記憶)上の実行ロードモジュール  
についても同様であった。

【0009】この問題を解決するのに、VLIWの命令  
形式を可変長としてnop命令の部分削除すること  
で、コードサイズを削減し、VLIWを解釈(デコー  
ド)・実行する際に、ハードウェア的に本来の形式に復  
元する方式が考えられる。しかし、この方式はハードウ  
ェア構成が極めて複雑になり、また性能も悪くなるた

め、実用的でない。しかも、分岐先の計算など、プログ  
ラムの制御構造の実現が極めて複雑になる。

【0010】本発明は上記事情を考慮してなされたもの  
でその目的は、従来VLIW上のnop命令で占められ  
ていたコード部分に起因する記憶領域の無駄を減らすこ  
とができるVLIW方式の計算機システム及びVLIW  
の解釈・実行方法を提供することにある。

【0011】本発明の他の目的は、従来VLIW上のn  
op命令で占められていたコード部分を変数または定数  
領域として利用でき、もって実行オブジェクトが使用す  
る主記憶領域を削減できるようにすることにある。本発  
明の更に他の目的は、ロードモジュールファイルのサイ  
ズを小さくできるようにすることにある。

【0012】

【課題を解決するための手段及び作用】本発明の第1の  
観点に係る構成は、並列動作可能な複数の演算器と、こ  
れら各演算器を個々に制御するための同数の独立した命  
令フィールドを持つVLIWであって、その一部に解釈  
・実行の必要のない無効命令フィールドがいずれである  
かを示す命令有効化情報を有し、次に解釈・実行すべき  
VLIWを保持するための命令レジスタと、この命令レ  
ジスタに保持されているVLIWの各命令フィールドに  
対応してそれぞれ設けられ、対応する命令フィールドの  
内容を解釈して当該フィールドに対応する演算器を制御  
するデコード手段であって、当該フィールドが上記命令  
有効化情報により無効命令フィールドであることが示さ  
れている場合には、当該フィールドの内容に従う対応す  
る演算器の動作を禁止するデコード手段と、上記命令有  
効化情報により無効であることが示されているVLIW  
上の命令フィールドに対して、演算オペランドとしてア  
クセスすることを許すメモリ管理手段とを備えたことを  
特徴とするものである。

【0013】上記第1の観点に係る構成において、命令  
レジスタに次に実行すべきVLIWがフェッチされた場  
合には、デコード手段は、そのVLIWの一部をなす命  
令有効化情報により有効命令フィールドであることが示  
されているフィールドの内容については、解釈(デコー  
ド)を行って、その解釈結果に従って対応する演算器の  
動作を制御する。また、命令有効化情報により無効命令  
フィールドであることが示されているフィールドの内容  
については、解釈を行わず、対応する演算器を動作させ  
ない。なお、解釈自体は行うものの、その解釈結果を無  
視し、対応する演算器を動作させないようにしても構わ  
ない。

【0014】このように、VLIW中の無効命令フィー  
ルドに対応する演算器は、当該VLIWの命令有効化情  
報に基づくデコード手段の制御により、その無効命令フ  
ィールドの内容に従って動作することが禁止される。この  
ため、その無効命令フィールドに従来とは異なってn  
op命令を配置せずに、その無効命令フィールドを演算

オペランドの領域として用いるようにしても、誤動作を招くことはない。この場合、従来VLIW上のnop命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができる。ここでVLIW中の無効命令フィールドを演算オペランドの領域として利用するには、ソースプログラムをコンパイルしてVLIWの列からなるオブジェクトプログラムを生成する際に、VLIW中の解釈・実行が不要な無効命令フィールドに定数または変数を割り当てるようにすればよい。

【0015】本発明の第2の観点に係る構成は、並列動作可能な複数の演算器を同数の独立した命令フィールドを持つVLIWにより制御するVLIW方式の計算機システムにおいて、VLIWの列からなるオブジェクト群をリンクしてロードモジュールを生成するリンク手段であって、当該ロードモジュールの生成時に、上記演算器を動作させる有効命令フィールドを除く無効命令フィールド部分を削除するリンク手段と、上記ロードモジュールから指定ページのVLIWの列を主記憶上にロードするためのデマンドページアクセス要求が発生した場合に、上記無効命令フィールド部分を含むVLIWの列を復元するデマンドページアクセス手段とを備えたことを特徴とするものである。

【0016】上記第2の観点に係る構成において、リンク手段によるリンク対象となるオブジェクト群を構成する各VLIWには、従来とは異なって、その一部に、解釈・実行の必要のない無効命令フィールドがいずれであるかを示す命令有効化情報が設けられている。

【0017】リンク手段は、各VLIW毎に、そのVLIWの命令有効化情報をもとに無効命令フィールドの位置を確認して、当該フィールドを取り除くことにより、無効命令フィールドが削除されたロードモジュールを生成する。これにより、ロードモジュールが置かれるファイルのサイズを削減することが可能となる。

【0018】一方、デマンドページアクセス手段は、デマンドページアクセス要求が発生すると、上記ロードモジュールから指定ページのVLIWの列を主記憶上にロードする。但し、当該ロードモジュールには無効命令フィールド部分は含まれていない。

【0019】そこでデマンドページアクセス手段は、指定ページのVLIWの列を主記憶上にロードする際には、対応するVLIWの命令有効化情報を参照して、当該VLIW上の前記無効命令フィールドの位置を検出し、その位置を飛ばして当該VLIW上の有効命令フィールドの内容を主記憶に配置する。これにより、VLIWが主記憶上に復元・配置される。

【0020】この他、命令有効化情報を持たない従来と同様の形式のVLIW、即ち無効命令フィールドにnop命令が設定されるVLIWについても、無効命令フィールド(nop命令)の削除によるロードモジュールのサイズの削減と、デマンドページアクセス時のVLIW

の復元を実現することが可能である。

【0021】このためには、まずリンク手段によるリンク時には、各VLIW毎に、そのVLIW中のnop命令を検出して、そのnop命令を除くことにより、nop命令が削除されたロードモジュールを生成すればよい。また、デマンドページアクセス時のVLIWの復元のために、リンク手段によるリンク時に、VLIWの列におけるnop命令(が設定される無効命令フィールド)とそれ以外の命令(が設定される有効命令フィールド)の配置を示す命令マップを生成してロードモジュールに付加しておく。

【0022】この場合、デマンドページアクセス手段は、ロードモジュールから指定ページのVLIWの列を主記憶上にロードする際に、当該ロードモジュールに付されている命令マップに従って、nop命令の位置を検出し、その位置にnop命令を配置することで、元のVLIWを主記憶上に復元・配置できる。

【0023】

【実施例】以下、本発明の実施例につき図面を参照して説明する。

【第1の実施例】図1は本発明の第1の実施例に係るVLIW方式の計算機システムの構成を示すブロック図、図2は同実施例で適用されるVLIWの命令フォーマットを示す図である。

【0024】まず、本実施例で適用されるVLIWの命令語長は128ビットである。この128ビットのVLIWは、図2に示すように、4つの32ビットの命令フィールドに分割される。

【0025】VLIWの1番目の命令フィールド(図面上で最も左側の命令フィールド)のうちの上位3ビット(図面上で最も左側の3ビット)はビットa2~a4からなる命令有効化情報aの設定フィールド(命令有効化フィールド)として用いられ、残りの29ビットはVLIWの1番目の命令(命令#1)を配置するのに用いられる。また、VLIWの2番目~4番目の命令フィールドは、VLIWの2番目~4番目の命令(命令#2~#4)を配置するのに用いられる。

【0026】命令有効化情報aを構成するビットa2~a4は、VLIW中のそれぞれ2番目~4番目の命令フィールドの設定内容が解釈・実行の必要のある有効な命令である(“1”の場合)か否か(“0”の場合)を示す。本実施例において、命令有効化情報aのビットai(i=2~4)が“0”の場合、VLIW中のi番目の命令フィールドは、変数または定数が設定される演算オペランド領域(以下、変数/定数領域と称する)として利用可能である。

【0027】図1の計算機システムにおいて、10はシステムの中核をなすCPU、20はVLIW方式のプログラム(オブジェクトプログラム)、オペランド等が格納される主記憶である。

【0028】CPU10は、主記憶20上の命令語群（VLIW群）の一部の写しが置かれる命令キャッシュ11と、この命令キャッシュ11からフェッチされたVLIWが保持される命令レジスタ12と、この命令レジスタ12に保持されたVLIW中の1番目～4番目の命令フィールドの設定内容の解釈（デコード）を行うデコーダ（DEC）13-1～13-4とを備えている。

【0029】デコーダ13-1は、VLIW中の1番目の命令フィールドの（命令有効化情報aを除く部分の）設定内容の解釈を無条件で行うものである。一方、他のデコーダ13-2～13-4は、VLIW中の2番目～4番目の命令フィールドの設定内容の解釈を、VLIW中の命令有効化情報aのビットa2～a4の状態（“1”であるか否か）に応じて行うものである。

【0030】CPU10はまた、デコーダ13-1～13-4の解釈結果に応じて演算を行う演算器14-1～14-4を持ち、命令実行と実行結果の書き込みを司る実行・書き込み部15と、主記憶20上のオペランド群の一部の写しが置かれるオペランドキャッシュ16とを備えている。このオペランドキャッシュ16のキャッシュライン長はVLIW長と同じ（ここでは128ビット）であるものとする。そして、一部の命令フィールドが変数/定数領域として利用されているVLIWがオペランドデータとして当該オペランドキャッシュ16に保持される場合、そのVLIWは、複数のキャッシュラインにまたがることなく、128ビットラインに配置されるものとする。

【0031】CPU10は更に、主記憶20のページ管理、（論理アドレスから物理アドレスへの）アドレス変換等を司るメモリ管理部17を備えている。メモリ管理部17は、論理アドレスを物理アドレスに高速に変換するためのアドレス変換バッファ機構であるTLB（translation lookaside buffer）171と、アクセス権の制御等を行うアクセス制御回路172とを備えている。

【0032】TLB171は、メモリ保護のためのアクセス権情報、論理アドレス（ここでは論理ページアドレス）をもとに生成されるタグ情報（アドレスタグ）、及び当該論理アドレスに対応する物理アドレス（ここでは物理ページアドレス）を含む情報を保持するための複数のエントリを有している。アクセス権情報は、対応するページに対してリードのみ許可するアクセス権RO、リードとライトの両方を許可するアクセス権RW及びアクセス権E等のいずれかを示すものである。アクセス権Eは、対応するページがVLIWの格納領域として用いられており、アクセス先となるVLIW中の領域に有効な命令がないならば、その領域を通常のオペランドアクセスと同様にアクセスできるという“実行可”を示す。

【0033】主記憶20は、メモリ管理部17によりページ単位で管理される。主記憶20内の領域は、例えばVLIWが格納されるページと、オペランドデータが格

納されるページ等に分けられる。但し、本実施例においては、後述するようにオペランドを含むVLIWが存在する。

【0034】次に、本発明の第1の実施例の動作を説明する。まず、CPU10において命令フェッチ要求が発生したものとする。もし、要求した命令語（VLIW）が命令キャッシュ11に存在するキャッシュヒット時には、その命令語（VLIW）が命令キャッシュ11から命令レジスタ12に高速に読み込まれる。

10 【0035】一方、要求した命令語（VLIW）が命令キャッシュ11に存在しないミスヒット時には、その命令語（VLIW）はメモリ管理部17を通して主記憶20からフェッチされ、命令キャッシュ11に保持された後、当該命令キャッシュ11から命令レジスタ12にフェッチされる。

20 【0036】なお、キャッシュヒット/ミスヒットの判定アルゴリズムや、命令キャッシュ11への命令語（VLIW）の登録処理等については、従来からよく知られており、しかも本発明に直接関係しないため説明を省略する。

30 【0037】命令レジスタ12に読み込まれたVLIW中の（1番目の命令フィールドの上位部分に設定された）3ビット命令有効化情報aのビットa2～a4は、デコーダ13-2～13-4に導かれる。また、当該VLIW中の1番目の命令フィールドの（命令有効化情報aを除く部分の）設定内容（命令#1）はデコーダ13-1に導かれ、2番目の命令フィールドの設定内容（命令#2または変数/定数）はデコーダ13-2に導かれる。更に、当該VLIW中の3番目の命令フィールドの設定内容（命令#3または変数/定数）はデコーダ13-3に導かれ、4番目の命令フィールドの設定内容（命令#4または変数/定数）はデコーダ13-4に導かれる。

【0038】デコーダ13-1は、命令レジスタ12から導かれたVLIWの1番目の命令フィールドの設定内容（命令#1）を無条件で解釈（デコード）し、その解釈結果に応じて実行・書き込み部15の演算器14-1を制御する。

40 【0039】一方、デコーダ13-2～13-4は、命令レジスタ12から導かれた命令有効化情報aのビットa2～a4に応じて、次のように動作する。まず、命令有効化情報aのビットa2～a4が“1”の場合には、デコーダ13-2～13-4は、命令レジスタ12から導かれたVLIWの2番目～4番目の命令フィールドに解釈・実行の必要な有効な命令（命令#2～#4）が設定されているものとして、その命令フィールドの設定内容（命令#2～#4）を解釈（デコード）し、その解釈結果に応じて実行・書き込み部15の演算器14-2～14-4を制御する。

50 【0040】次に、命令有効化情報aのビットa2～a4が“0”の場合には、デコーダ13-2～13-4は、命

令レジスタ12から導かれたVLIWの2番目～4番目の命令フィールドには解釈・実行の必要な有効な命令は設定されておらず、代わりに変数/定数が設定されている可能性があるものとして、その命令フィールドを無効扱い（無効命令フィールド、不要命令フィールド）とし、その命令フィールドの設定内容の解釈を控えて、演算器14-2～14-4を動作させない。なお、解釈（デコード）動作自体は行っても、その解釈結果を無視することで、対応する演算機を動作させないようにしても構わない。

【0041】以上のデコーダ13-2～13-4の動作により、VLIWの一部（ここでは1番目の命令フィールドの先頭部分）に設けられた命令有効化情報aのビットa2～a4で命令の無効化が指定されているならば、VLIWの対応する2番目～4番目の命令フィールドには、どのような32ビットデータを配置しても、そのフィールドに対応する演算器14-2～14-4が誤動作することはない。

【0042】したがって、従来であればnop命令を配置する必要のあるVLIW中の命令フィールドを変数または定数の設定領域（変数/定数領域）として利用することができる。但し、1番目の命令フィールドの設定内容は、デコーダ13-1により常時解釈されて、実行されるため、実行が不要な場合には、当該フィールドに従来通りnop命令を配置する必要がある。

【0043】なお、本実施例においては、1つのVLIW中に配置される命令の長さが、29ビットと32ビットの2種類ある（図2参照）。しかし、VLIW自体は固定長であり、しかも29ビット長の命令（命令#1）は常にVLIWの1番目の命令フィールドに設定され、32ビット長の命令（命令#2～#4）は常にVLIWの2番目～4番目の命令フィールドに設定されることから、何ら問題とはならず、ハードウェア構成も（nop命令を削除してVLIWを可変長とする方式とは異なつて）図1に示したように簡単である。また、VLIWの1番目の命令フィールドに対応する演算器14-1で実行可能な命令の種類は、他の命令フィールドに対応する演算器14-2～14-4で実行可能な命令の種類より少なくなるが、演算器14-1で実行できない命令をVLIWの2番目以降の命令フィールドに配置して演算器14-2～14-4で実行させることで対処できる。

【0044】さて、デコーダ13-1～13-4の少なくとも1つで命令の解釈が行われて、実行・書き込み部15内の対応する演算器が動作する際に、その動作に必要なオペランド（変数/定数）を（例えばオペランドキャッシュ16から）リードする必要がある、或いは演算器の動作が行われた後、その演算結果を（例えばオペランドキャッシュ16に）ライトする必要があるものとする。この場合、実行・書き込み部15からメモリ管理部17に対してオペランドアクセス（オペランドデータアクセ

ス）要求が発生する。

【0045】するとメモリ管理部17では、図3のフローチャートに従って以下に述べるオペランドアクセス処理が行われる。まず、メモリ管理部17内のアクセス制御回路172は、実行・書き込み部15からのオペランドアクセス要求を受け取ると、TLB171を参照し、その要求先のアドレス（論理ページアドレス）に対応する物理ページアドレスが登録されているエントリを検索する。

10 【0046】もし、TLB171に目的とするエントリが存在するヒット時であれば、アクセス制御回路172は当該エントリ中の物理ページアドレスを用いて論理アドレスから物理アドレスへの周知のアドレス変換を高速に行うと共に、当該エントリ中のアクセス権情報を得る。

【0047】一方、TLB171に目的とするエントリが存在しないミスヒット時であれば、アクセス制御回路172は、主記憶20の所定領域に置かれているページテーブル（図示せず）をアクセスし、当該ページテーブルを用いて論理アドレスから物理アドレスへの周知のアドレス変換を行う。このページテーブルには、OS（オペレーティングシステム）の管理により、各ページ毎のアクセス権情報が設定されている。このアクセス権情報としては、リードのみ許可するアクセス権RO、リードとライトの両方を許可するアクセス権RWなど従来からよく知られているメモリ保護のためのアクセス権の他に、本実施例特有のアクセス権Eが用意されている。このアクセス権Eは、前記したように、対応するページがVLIWの格納領域として用いられており、アクセス先となるVLIW中の領域に有効な命令がないならば、その領域を通常のオペランドアクセスと同様にアクセスできるという“実行可”を示すものである。本実施例において、RO=0、RW=1、E=2である。

【0048】アクセス制御回路172は、ページテーブルを用いたアドレス変換を行うと、当該テーブルから得られる該当ページのアクセス権情報、要求先の論理アドレス（ここでは論理ページアドレス）をもとに生成されるタグ情報（アドレスタグ）、及びアドレス変換結果である物理ページアドレスを含む情報をTLB171内のエントリに登録する周知の登録動作を行う。

【0049】さてアクセス制御回路172は、TLB171またはページテーブルを用いたアドレス変換により、要求先アドレス（論理アドレス）に対応する物理アドレスとアクセス権情報を得ると（ステップS1）、オペランドのリード時であれば、アクセス権情報の示すアクセス権がRO或いはRWの場合、オペランドのライト時であればアクセス権情報の示すアクセス権がRWの場合に、取得した物理アドレスをもとにオペランドキャッシュ16を対象とするオペランドアクセスを実行し、リード時であればリードしたオペランドを実行・書き込み



部15に渡す。この場合の動作は、従来と何ら変わらない。なお、オペランドキャッシュ16に目的とするデータが存在しないミスヒット時には、主記憶20がアクセスされることはいうまでもない。

【0050】これに対し、アクセス権情報の示すアクセス権がEの場合、即ち実行可の場合には(ステップS2)、アクセス制御回路172は、取得した物理アドレスをもとにオペランドキャッシュ16の該当するキャッシュラインをリードアクセスする(ステップS3)。アクセス権がE(実行可)の場合、アクセスしたキャッシュラインの情報はVLIWである。

【0051】アクセス制御回路172は、アクセスしたキャッシュラインの情報(VLIW)の先頭の32ビット、即ちVLIWの1番目の命令フィールドから命令有効化情報aを取り出す(ステップS4)。なお、ミスヒット時には、命令有効化情報aの取り出しは、主記憶20を対象に行われる。

【0052】アクセス制御回路172は、命令有効化情報aの取り出しを行うと、オペランドアクセスすべきフィールド(データ語部分)に有効な命令が存在するか否かを、当該命令有効化情報aの対応ビット(の状態)をもとに調べる(ステップS5)。

【0053】もし、有効な命令が存在するならば、アクセス制御回路172は、不正アクセス(アクセスエラー)として、要求されたオペランドアクセスを行わずに当該命令を保護し、オペランドキャッシュ16にその旨を通知する(ステップS6)。

【0054】これに対し、有効な命令が存在しないならば、アクセス制御回路172は、通常のオペランドデータと同様に、オペランドアクセスすべきフィールド(データ語部分)を対象とするアクセスを行う(ステップS7)。

【0055】このように、VLIW中2番目～4番目の命令フィールドのうち、命令の依存関係で従来であればnop命令を配置しなければならない命令フィールドを変数/定数領域として利用することで、従来VLIW上のnop命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができる。しかも、変数/定数領域として利用する命令フィールドの設定内容は、命令有効化情報aの対応ビットの無効化指定により、対応するデコーダ13-i(iは2～4のいずれか)で解釈されず、無効化扱いされるため、演算器14-iが誤動作する虞はない。

【0056】図4は図1の計算機システムで適用される、図2に示したような形式のVLIW群を含むオブジェクトプログラムをソースプログラムから生成するコンパイラの構成を示す。

【0057】図4のコンパイラは、ソースプログラム31をコンパイルして従来と同様の形式のVLIW群を含む第1のオブジェクトプログラム32を生成する第1の

コンパイル部41と、この第1のオブジェクトプログラム32から図2の形式の第2のオブジェクトプログラム33を生成する第2のコンパイル部42とから構成される。

【0058】次に、図4の構成のコンパイラの動作を、図5及び図6のフローチャートを参照して説明する。まず、第1のコンパイル部41は、ソースプログラム31のソースコード列から、各ソースコード毎に、逐次実行型の命令または命令群を作成し、VLIWへの再配置前の命令列(オブジェクトコード列)を生成する(ステップS11)。

【0059】次に第1のコンパイル部41は、生成した再配置前の命令列を4つの32ビット命令フィールドを持つVLIW型に最適化、再配置し、各命令間の依存関係により演算器を動作させる命令を配置できない命令フィールドにはnop命令を配置することで、従来と同様の形式のVLIW群を含む第1のオブジェクトプログラム32を生成する(ステップS12)。但し、VLIWの1番目の命令フィールドには、先頭の3ビットを除く領域に29ビット命令またはnop命令が配置される点で、従来と少し異なる。なお、29ビット命令は、上位の3ビットに無関係に演算指定可能な32ビット命令と等価である。

【0060】上記のようにして第1のコンパイル部41により生成される第1のオブジェクトプログラム32は、(従来と同様の形式の)VLIW群からなるコード部321と、コード部321上の命令から参照されるデータ(変数/定数)の集合であるデータ部322から構成される。このデータ部322のデータは、そのデータが置かれる論理的な位置を示す情報(ラベル)と関連付けられている。

【0061】第1のコンパイル部41により第1のオブジェクトプログラム32が生成されると、第2のコンパイル部42が起動される。すると第2のコンパイル部42は、第1のオブジェクトプログラム32のコード部321の先頭VLIWの2番目の命令フィールドを参照して(ステップS13)、そこにnop命令が配置されているか否かをチェックする(ステップS14)。

【0062】第2のコンパイル部42は、nop命令が配置されている命令フィールドを検出した場合、そのフィールドを変数/定数領域として利用するならば(ステップS15)、そのフィールドに第1のオブジェクトプログラム32のデータ部322上の任意のデータを(nop命令に代えて)配置する(ステップS16)。

【0063】次に第2のコンパイル部42は、コード部321をスキャンして、配置したデータを参照する命令を全て探し、これら各命令に設定されている当該データの参照先を示すラベルを、ステップS16で配置した命令フィールドの位置を示すラベルにそれぞれ書き換える(ステップS17)。なお、第1のコンパイル部41に

より第1のオブジェクトプログラム32を生成する際に、コード部321上の各命令のうち、データ部322上のデータを参照する命令の位置情報を、当該データと対応させてテーブル等に保持しておくならば、上記のステップS17でのラベル書き換えの対象となる命令を簡単に探すことができる。

【0064】第2のコンパイル部42は、ステップS17を実行すると、現在注目しているVLIW（現VLIW）の最後の命令フィールドまで終了したか否かをチェックする（ステップS18）。このステップS18のチェックは、参照した命令フィールドにnop命令が配置されていない場合と、参照した命令フィールドを変数／定数領域として利用しない場合にも行われる。

【0065】もし、現VLIWの最後の命令フィールドまで終了していなければ、第2のコンパイル部42は、現VLIWの次の命令フィールドを参照して（ステップS19）、ステップS14に戻る。

【0066】一方、現VLIWの最後の命令フィールドまで終了しているならば、第2のコンパイル部42は、そのVLIWの1番目の命令フィールドの上位3ビット部分に、2番目～4番目の命令フィールドの内容に応じた命令有効化情報aを設定する（ステップS20）。

【0067】第2のコンパイル部42は、命令有効化情報aの設定処理を行うと、コード部321の最後のVLIWまで終了したか否かをチェックする（ステップS21）。

【0068】もし、コード部321の最後のVLIWまで終了していなければ、第2のコンパイル部42は、コード部321上の次のVLIWの2番目の命令フィールドを参照して（ステップS22）、ステップS14に戻る。

【0069】一方、コード部321の最後のVLIWまで終了しているならば、第2のコンパイル部42はコンパイル処理を終了する。これにより、図2の形式のVLIWの群からなるコード部33と、コード部321上の命令から参照されるデータ（変数／定数）の集合であるデータ部322から構成される、第2のオブジェクトプログラム33が生成される。この第2のオブジェクトプログラム33が、図1の計算機システムで用いられる。

【0070】以上、従来であればnop命令を配置する必要のあるVLIW中の命令フィールドを変数／定数領域として利用できるようにすることで、実行オブジェクトが使用する主記憶20の領域の削減を図った第1の実施例につき説明した。

【0071】なお、上記第1の実施例では、128ビット長のVLIWの1番目の32ビット長命令フィールドの一部を命令有効化情報aとして用いているが、4つの32ビット長の命令フィールドと命令有効化情報aからなる131ビット長のVLIWを用いるようにしても構わない。いずれの場合も、VLIWの一部を命令有効化

情報aとして用いることに変わりはない。また、命令フィールドの長さ、命令フィールドの数も、上記第1の実施例に限るものでないことは勿論である。

【0072】また、上記第1の実施例では、図4のフローチャートに従うコンパイル処理のステップS12において、従来と同様の形式のVLIWの列を含む第1のオブジェクトプログラム32を生成し、命令有効化情報aの設定はステップS20で行うものとしたが、ステップS12の段階で、無効命令フィールドにnop命令とは別の無意味なコードを配置するとか、或いは当該フィールドを空きフィールドとし、VLIWの先頭命令フィールドに命令有効化情報aを配置するようにしてもよい。

【第2の実施例】次に、VLIW方式のロードモジュールファイルのサイズを小さくできるようにすることで、当該ファイルの占める補助記憶領域（ディスク領域）の削減を図るようにした第2の実施例につき、図面を参照して説明する。

【0073】図7はロードモジュールを生成して補助記憶に格納しておき、デマンドページアクセス要求が発生した場合に、要求されたロードモジュールのページを主記憶に展開するための構成を示す。

【0074】同図において、51はリンク手段としてのリンケージエディタ、52はオペレーティングシステム（OS）の仮想記憶管理部、53は主記憶である。リンケージエディタ51は、VLIW方式のオブジェクト群（オブジェクトプログラム群）61をリンクして計算機システム（内のCPU）で実行可能な圧縮されたロードモジュール62を生成するものである。本実施例で適用されるオブジェクト群61は、前記第1の実施例で適用した図2の形式のVLIWの列、即ち4つの4バイト命令フィールドから構成され（VLIW長＝16バイト）、1番目の命令フィールドの先頭部分に命令有効化情報aを持つVLIWの列からなるものとする。但し、命令有効化情報a内のビットai（i＝2～4）により（解釈・実行の必要がないものとして）無効（不要）指定される命令フィールドは、前記第1の実施例とは異なっており、変数／定数領域として用いられず空き領域として位置付けられるものとする。

【0075】ロードモジュール62は、補助記憶、例えばディスク装置に置かれるロードモジュールファイル63に格納されるもので、ページ数分のコード部621と、ページ数分のインデックス部622とから構成される。

【0076】各コード部621は、対応するページのVLIW列を圧縮した命令群（具体的にはVLIW列から無効命令フィールド部分が削除された残りの命令群）が格納される圧縮コード領域として用いられるものである。

【0077】各インデックス部622は、対応するページのコード部621の先頭位置を（ロードモジュールフ

ファイル63内の相対位置で示す情報（以下、ページインデックスと称する）を格納しておくためのものである。

【0078】仮想記憶管理部52は、デマンドページアクセス要求が発生した場合に、ロードモジュールファイル63内のロードモジュール62から対応するページのVLIW列を復元して主記憶53に配置するものである。

【0079】次に、図7の構成の動作を、(1)リンク処理と(2)デマンドページアクセス処理に分けて、図8乃至図10のフローチャートを参照して順に説明する。

#### (1) リンク処理

まず、リンケージエディタ51によるリンク処理について説明する。

【0080】リンケージエディタ51は、オブジェクト群61をリンクしてロードモジュール62を生成する際には、必要とするページ数分のインデックス部622をロードモジュールファイル62内に確保する（ステップS31）。

【0081】次にリンケージエディタ51は、ロードモジュール62の先頭ページ用のコード部621をロードモジュールファイル62内に確保し、そのコード部621の先頭のロードモジュールファイル63内相対位置を示すページインデックスを、対象ページに固有のインデックス部622に設定する（ステップS32）。

【0082】次にリンケージエディタ51は、オブジェクト群61から対象ページの先頭VLIWを取り出す（ステップS33）。そしてリンケージエディタ51は、取り出したVLIWの1番目の命令フィールドの先頭部分に設定されている命令有効化情報aを参照し、解釈・実行の必要のない無効（不要）フィールドであることが示されている命令フィールド部分を当該VLIWから取り除いて、コード部621に例えば1命令フィールド単位で格納する（ステップS34）。

【0083】これにより、命令有効化情報a（中のビットa2～a4）が例えば“010”であるVLIWの場合には、当該VLIW中の2番目と4番目の命令フィールド部分が削除され、コード部621には、残りの1番目と3番目の命令フィールドの内容だけが格納される。同様に、命令有効化情報a（中のビットa2～a4）が例えば“101”であるVLIWの場合には、当該VLIW中の3番目の命令フィールド部分が削除され、コード部621には、残りの1番目と2番目と4番目の命令フィールドの内容だけが格納される。この様子を図11(a)に示す。

【0084】次にリンケージエディタ51は、対象ページの最後のVLIWまで終了したか否か、即ちページエンドとなったか否かをチェックする（ステップS35）。ここで、ページサイズを16Kバイトとすると、

1VLIWが4つの4バイト命令フィールドから構成されていることから（VLIW長=16バイト）、1ページのVLIW数は1024である。したがってリンケージエディタ51は、1024個のVLIWを処理する毎に、ページエンドを判断する。

【0085】もし、ページエンドとなっていないならば、リンケージエディタ51は、オブジェクト群61から同じページの次のVLIWを取り出して（ステップS36）、ステップS34に戻る。

【0086】一方、ページエンドとなっているならば、リンケージエディタ51は、必要なページ数分の処理を終了したか否かをチェックする（ステップS37）。もし、必要なページ数分の処理を終了していないならば、リンケージエディタ51は、次のページ用のコード部621をロードモジュールファイル62内に確保し、そのコード部621の先頭のロードモジュールファイル63内相対位置を示すページインデックスを、対象ページに固有のインデックス部622に設定する（ステップS38）。そしてリンケージエディタ51は、ステップS33に戻る。

【0087】一方、必要なページ数分の処理を終了しているならば、リンケージエディタ51はリンク処理を終了する。このようにして、ロードモジュールファイル63には、各VLIWから解釈・実行の必要のない無効命令フィールド部分が取り除かれた命令群（圧縮されたVLIW列）が格納されたコード部621と、当該コード部621の先頭のロードモジュールファイル63内相対位置を示すインデックス部622とをページ数分持つロードモジュール62が生成される。このロードモジュール62は、VLIW中の無効命令フィールド部分を含めないことから、ロードモジュールファイル63のサイズは従来に比べて小さくて済む。

#### (2) デマンドページアクセス処理

次に、仮想記憶管理部52によるデマンドページアクセス処理について説明する。

【0088】計算機システムの動作中にデマンドページアクセス要求が発生すると、ロードモジュールファイル63をアクセスして要求されたページのVLIW列を主記憶53上の指定領域に配置（ロード）する動作が、仮想記憶管理部52により次のように行われる。

【0089】まず仮想記憶管理部52は、主記憶53上の命令配置先を指し示すポインタ（配置先ポインタ）の値として、上記指定領域の先頭位置を設定する（ステップS41）。

【0090】次に仮想記憶管理部52は、ロードモジュール62中の要求されたページに固有のインデックス部622を参照し、当該ページの（命令群が格納されている）コード部621の先頭位置を認識する（ステップS42）。

【0091】次に仮想記憶管理部52は、そのコード部

621から先頭の命令、即ちVLIWの1番目のフィールドの内容を取り出す(ステップS43)。次に仮想記憶管理部52は、取り出した命令(VLIWの1番目のフィールドの内容)を配置先ポインタの指す主記憶53の4バイト領域に配置(ロード)して(ステップS44)、当該ポインタを4バイト分進める(ステップS45)。

【0092】次に仮想記憶管理部52は、命令有効化情報a中のビットaiを指定するための変数iを初期値2に設定した後(ステップS46)、上記取り出した命令の先頭部分に設定されている命令有効化情報a中のビットaiを参照して(ステップS47)、当該aiが“1”であるか否かをチェックする(ステップS48)。

【0093】もし、ai=1であれば、仮想記憶管理部52はコード部621から次の命令を取り出して(ステップS49)、配置先ポインタの指す主記憶53の4バイト領域に配置し(ステップS50)、しかる後に当該ポインタを4バイト分進めると共にiを+1する(ステップS51、S52)。

【0094】一方、ai=0であれば、仮想記憶管理部52は、ステップS49、S50をスキップして、配置先ポインタを4バイト分進めると共にiを+1する(ステップS51、S52)。これにより、主記憶53上に、無効命令フィールドの領域が確保されることになる。

【0095】仮想記憶管理部52は、ステップS51、S52を実行すると、iの値が4を越したか否かをチェックする(ステップS53)。もし、iが4を越していないならば、仮想記憶管理部52はステップS47に戻って、命令有効化情報a中の次のビット(ビットai)を参照し、その値により、ステップS48~S53またはステップS48、S51、S53を実行する。

【0096】一方、iが4を越しているならば、仮想記憶管理部52は、主記憶53上に1VLIWが復元・配置できたものと判断し、要求されたページの最後のVLIWまで配置できたか否か、即ちページエンドとなったか否かをチェックする(ステップS54)。

【0097】もし、ページエンドとなっていないならば、仮想記憶管理部52は、コード部621から次の命令、即ち次のVLIWの1番目のフィールドの内容を取り出して(ステップS55)、ステップS44に戻る。

【0098】一方、ページエンドとなっているならば、仮想記憶管理部52は、要求されたページのVLIW列が全て主記憶53に復元・配置できたものとして、デマンドページアクセス処理を終了する。

【0099】以上のデマンドページアクセス処理によれば、VLIWの無効フィールドの内容が削除されて格納されているコード部621の内容から、ステップS43またはステップS55で取り出される命令(VLIWの

1番目の命令フィールドの内容)の先頭部分にある命令有効化情報aに応じて、元のVLIWが正しく復元されて主記憶53に配置(ロード)される。

【0100】例えば、ステップS43またはステップS55で取り出される命令の先頭部分にある命令有効化情報a(中のビットa2~a4)が“010”の場合には、主記憶53上で当該命令(命令#1)から1命令分スキップして無効命令フィールドを確保した状態で、コード部621上の次の命令(命令#3)を配置し、しかる後に更に1命令分スキップして無効命令フィールドを確保することで、元のVLIWが復元・配置される。同様に、VLIWの先頭命令の先頭部分にある命令有効化情報a(中のビットa2~a4)が“101”の場合には、主記憶53上で当該命令(命令#1)の後続位置に、コード部621上の次の命令(命令#2)を配置し、しかる後に1命令分スキップして無効命令フィールドを確保した状態で、コード部621上の更に次の命令(命令#4)を配置することで、元のVLIWが復元・配置される。この様子を図11(b)に示す。

20 【0101】以上、リンク処理時に無効(不要)命令部分(命令フィールド)が取り除かれたロードモジュールを生成することで、ロードモジュールファイルの占める補助記憶領域(ディスク領域)を削減でき、しかもデマンドページアクセス要求に従うプログラムロード時に元のVLIW列に復元することで、正常なプログラム処理が行えるようにした第2の実施例につき説明した。

【第3の実施例】次に、前記第2の実施例で述べたロードモジュールサイズを削減する技術を、従来と同様の形式のVLIWの列、即ち無効命令フィールドにnop命令が設定されるVLIWの列を扱う計算機システムに応用した第3の実施例につき、図12を参照して説明する。

30 【0102】図12において、71はリンケージエディタ、72は仮想記憶管理部、73は主記憶である。リンケージエディタ71は、VLIW方式のオブジェクト群(オブジェクトプログラム群)81をリンクして計算機システム(内のCPU)で実行可能な圧縮されたロードモジュール82を生成するものである。本実施例で適用されるオブジェクト群81は、従来と同様の形式のVLIW列からなるものとする。

40 【0103】ロードモジュール82は、補助記憶、例えばディスク装置に置かれるロードモジュールファイル83に格納されるもので、ページ数分のコード部821と、ページ数分のインデックス部822とから構成される。

50 【0104】各コード部821は、対応するページのVLIW列を圧縮した命令群(具体的にはVLIW列からnop命令が削除された残りの命令群)が格納される圧縮コード領域821aと、圧縮前のVLIW列におけるnop命令とそれ以外の命令の配置を示すマップ(命令

マップ)が格納される命令マップ領域821bとから構成される。

【0105】各インデックス部822は、対応するページのコード部821の先頭位置を(ロードモジュール8内の相対位置で)示す情報(ページインデックス)を格納しておくためのものである。

【0106】仮想記憶管理部72は、デマンドページアクセス要求が発生した場合に、ロードモジュールファイル83内のロードモジュール82から対応するページのVLIW列を復元して主記憶73に配置するものである。

【0107】次に、図12の構成の動作を説明する。まず、リンケージエディタ71は、オブジェクト群81をリンクしてロードモジュール82を生成する際には、先頭ページから順に、そのページに割り当てるロードモジュールファイル83内のコード部821を確保する。

【0108】リンケージエディタ71は、1ページ分のコード部821を確保する毎に、オブジェクト群81における、そのページのVLIW列について、先頭VLIWから順にnop命令を取り除きながら、当該コード部821の圧縮コード領域821aの先頭から格納していく。

【0109】リンケージエディタ71はまた、上記したコード部821の圧縮コード領域821aへのnop命令を除く命令格納と並行して、該当するページのVLIW毎に、先頭VLIWから順に、VLIW上のnop命令とそれ以外の命令の配置を示す情報を、当該コード部821の先頭に設けられた命令マップ領域821bに格納していく。

【0110】この命令マップ領域821bのサイズは、1VLIWが4つの4バイト命令フィールドから構成され(VLIW長=16バイト)、ページサイズを16Kバイトとすると、1ページのVLIW数は1024であることから、VLIW上のnop命令とそれ以外の命令の配置を1命令(1命令フィールド)当り1ビットで示すならば、1024×4ビットの固定サイズとなる。

【0111】リンケージエディタ71は更に、上記確保したコード部821の先頭のロードモジュールファイル83内相対位置を示すページインデックスを、対象となっているページに固有のインデックス部822に格納する。

【0112】このようにして、ロードモジュールファイル83には、nop命令が取り除かれた命令群が格納されたコード部821と、当該コード部821の先頭のロードモジュールファイル83内相対位置を示すインデックス部822とをページ数分持つロードモジュール82が生成される。このロードモジュール82は、nop命令が取り除かれていることから、ロードモジュールファイル83のサイズは従来に比べて小さくて済む。

【0113】さて、計算機システムの動作中にデマンド

ページアクセス要求が発生すると、ロードモジュールファイル83をアクセスして要求されたページのVLIW列を主記憶73に配置する動作が、仮想記憶管理部72により次のように行われる。

【0114】まず仮想記憶管理部72は、ロードモジュール82中の要求されたページに固有のインデックス部822を参照し、当該ページの(命令群が格納されている)コード部821の先頭位置を認識する。

【0115】次に仮想記憶管理部72は、そのコード部821の先頭位置から始まる命令マップ領域821bに格納されている命令マップを先頭ビットから順に参照しながら、その命令マップ領域821bと対をなす圧縮コード領域821aの先頭から順に命令を取り出して主記憶73に配置(ロード)する。このとき、参照したビットが“0”のときは、仮想記憶管理部72はnop命令が取り除かれているものと判断し、圧縮コード領域821aから取り出した命令に先行して、nop命令を主記憶73に配置する。そして、参照したビットが“0”から“1”に変わったところで、圧縮コード領域821aから取り出した命令をnop命令に後続して主記憶73に配置する。

【0116】このようにして、要求されたページのVLIW列が正しく復元されて主記憶73に配置される。以上、従来と同様の形式のVLIW列を適用していても、リンク処理時にnop命令が取り除かれたロードモジュールを生成することで、ロードモジュールファイルの占める補助記憶領域(ディスク領域)を削減でき、しかもデマンドページアクセス要求に従うプログラムロード時に元のVLIW列に復元することで、正常なプログラム処理が行えるようにした第3の実施例につき説明した。

【0117】

【発明の効果】以上詳述したように本発明によれば、従来VLIW上のnop命令で占められていたコード部分を変数または定数領域として利用できるようになり、これにより実行オブジェクトが使用する主記憶領域を削減することができる。

【0118】また、本発明によれば、従来VLIW上のnop命令で占められていた無効(不要)命令部分(命令フィールド)が取り除かれたロードモジュールを生成することで、ロードモジュールファイルの占める補助記憶領域を削減でき、しかもデマンドページアクセス要求に従うプログラムロード時に元のVLIW列に復元することで、正常なプログラム処理が行える。このよう本発明によれば、従来VLIW上のnop命令で占められていたコード部分に起因する記憶領域の無駄を減らすことができる。

【図面の簡単な説明】

【図1】本発明の第1の実施例に係るVLIW方式の計算機システムの構成を示すブロック図。

【図2】同実施例で適用されるVLIWの命令フォーマ

ットを示す図。

【図3】図1中のメモリ管理部17によるオペランドアクセス処理を説明するためのフローチャート。

【図4】図1の計算機システムで適用される、図2に示す形式のVLIW群を含むオブジェクトプログラムをソースプログラムから生成するコンパイラの構成を示す図。

【図5】図4の構成のコンパイラによるコンパイル処理を説明するためのフローチャートの一部を示す図。

【図6】図4の構成のコンパイラによるコンパイル処理を説明するためのフローチャートの残りを示す図。

【図7】VLIW方式のロードモジュールファイルのサイズを小さくできるようにすることで、当該ファイルの占める補助記憶領域の削減を図るようにした第2の実施例を示す図。

【図8】図7中のリンケージエディタ51によるリンク処理を説明するためのフローチャート。

【図9】図7中の仮想記憶管理部52によるデマンドページアクセス処理を説明するためのフローチャートの一部を示す図。

【図10】図7中の仮想記憶管理部52によるデマンドページアクセス処理を説明するためのフローチャートの残りを示す図。

【図11】図7の構成の動作を説明するための図であり、図11(a)はリンク処理時のVLIWからの無効

命令フィールド削除の様子を、図11(b)はデマンドページアクセス処理時のVLIW復元の様子を示す図。

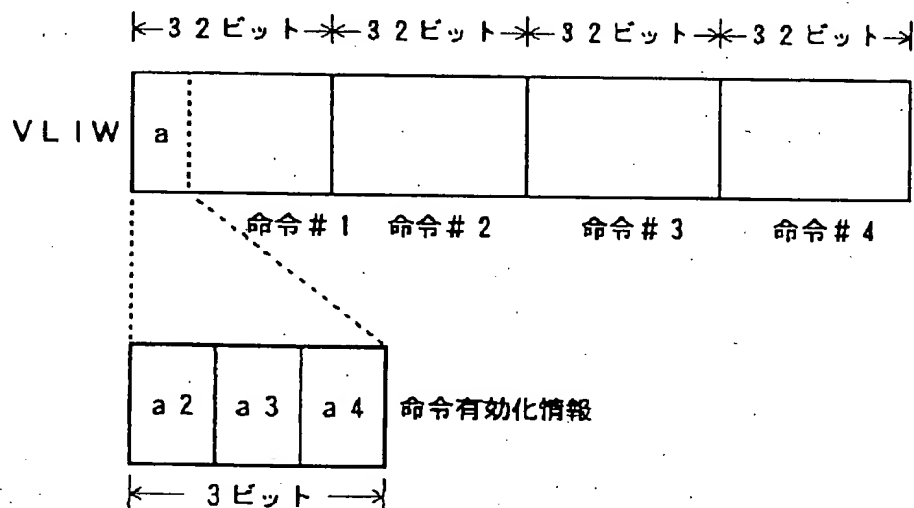
【図12】VLIW方式のロードモジュールファイルのサイズを小さくできるようにすることで、当該ファイルの占める補助記憶領域の削減を図るようにした第3の実施例を示す図。

【図13】従来のVLIW方式の計算機システムの要部構成を示す図。

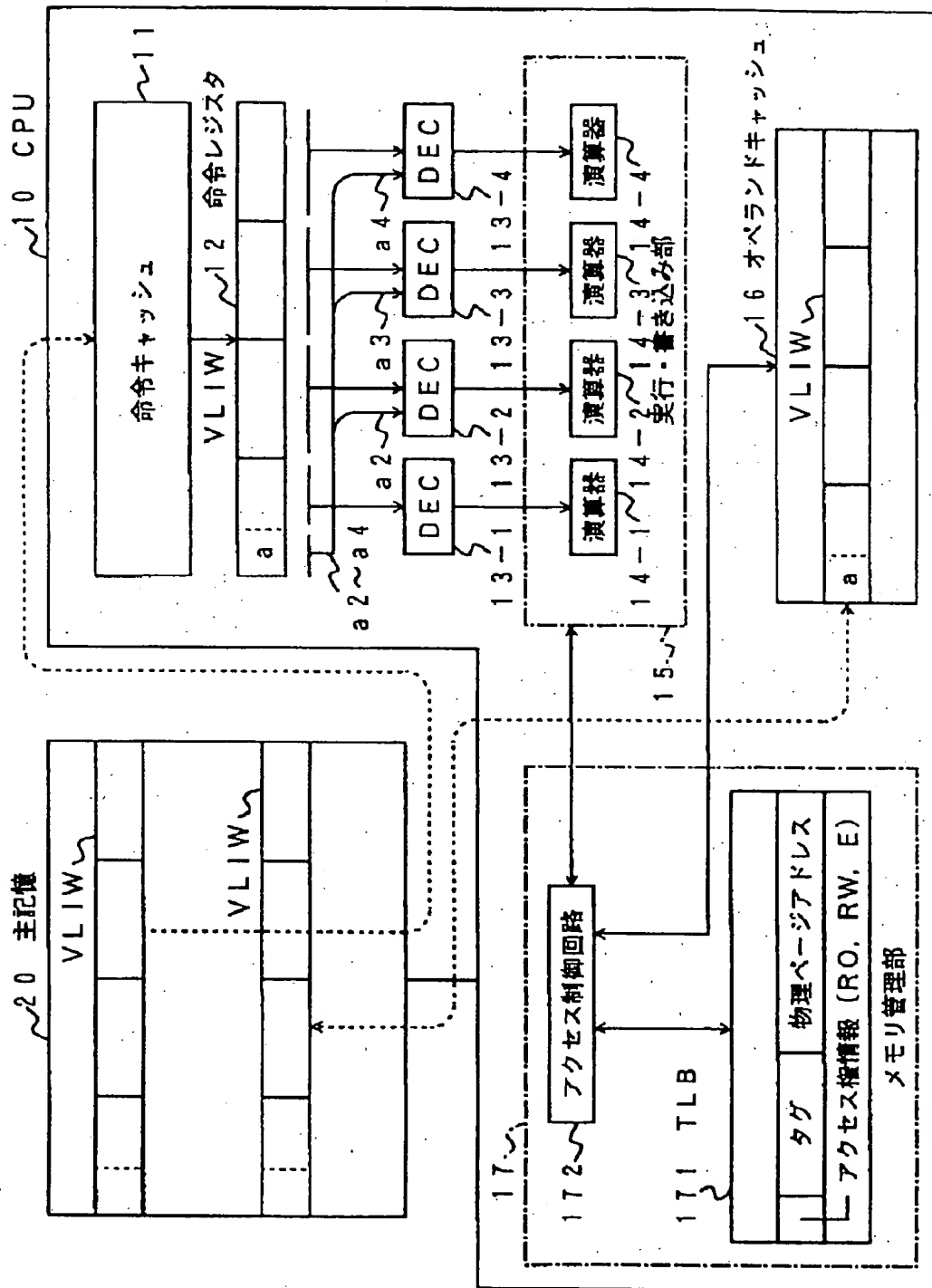
【符号の説明】

- 10 10…CPU、11…命令キャッシュ、12…命令レジスタ、13-1～13-4…デコード(DEC)、14-1～14-4…演算器、15…実行・書き込み部、16…オペランドキャッシュ、17…メモリ管理部、171…TLB、172…アクセス制御回路、20…主記憶、31…ソースプログラム、32…第1のオブジェクトプログラム、33…第2のオブジェクトプログラム、41…第1のコンパイル部、42…第2のコンパイル部、321、331…コード部、322、332…データ部、51、71…リンケージエディタ(リンク手段)、52、72…仮想記憶管理部(デマンドページアクセス手段)、53、73…主記憶、61、81…オブジェクト群、62、82…ロードモジュール、63、83…ロードモジュールファイル、621、821…コード部、622、822…インデックス部、821a…圧縮コード領域、821b…命令マップ領域。

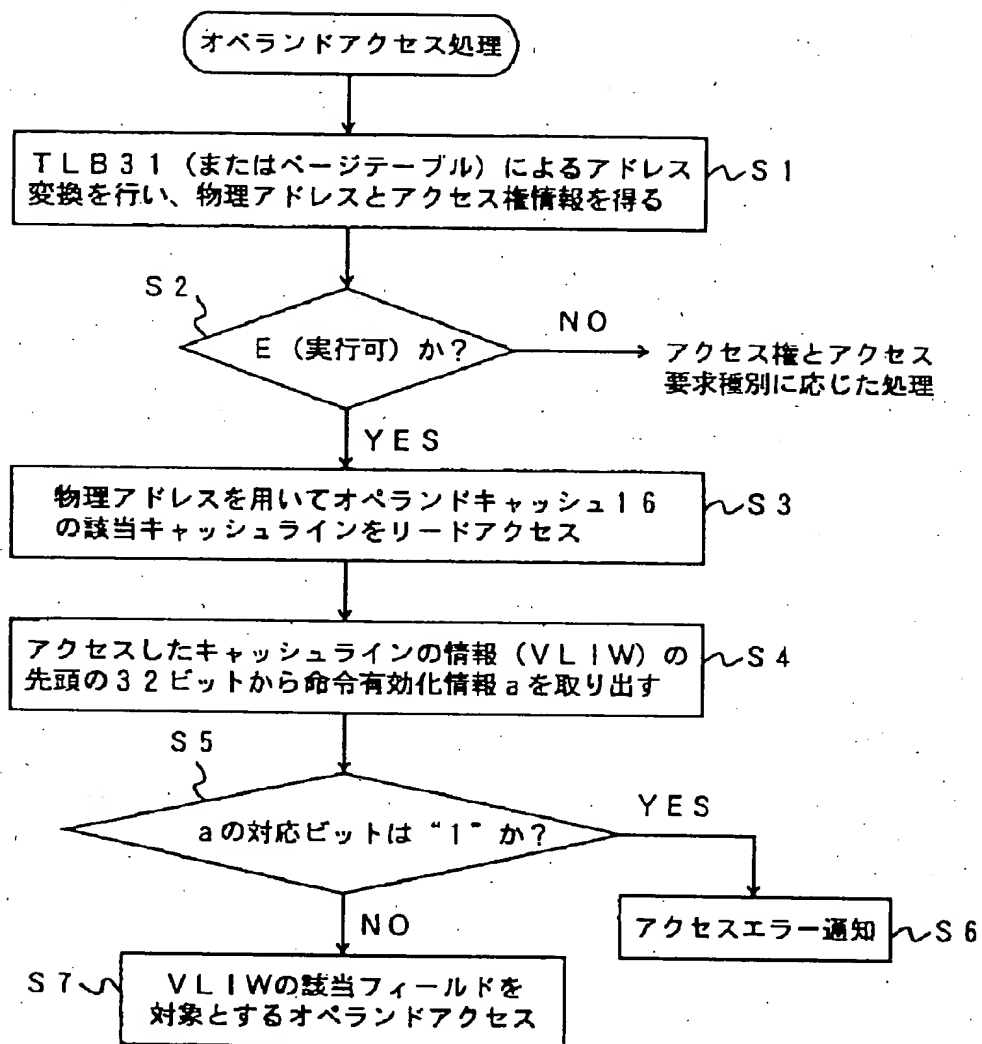
【図2】



【図1】

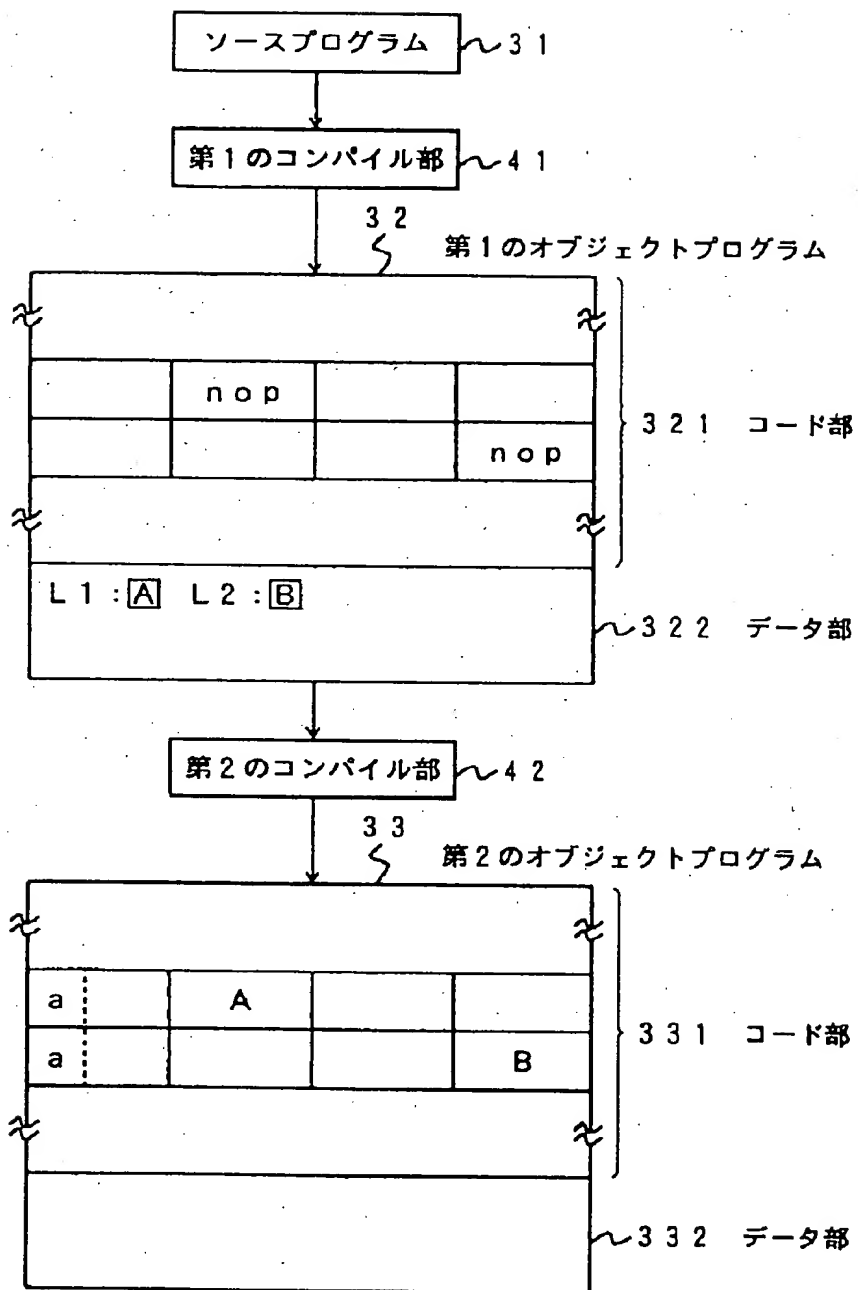


【図3】

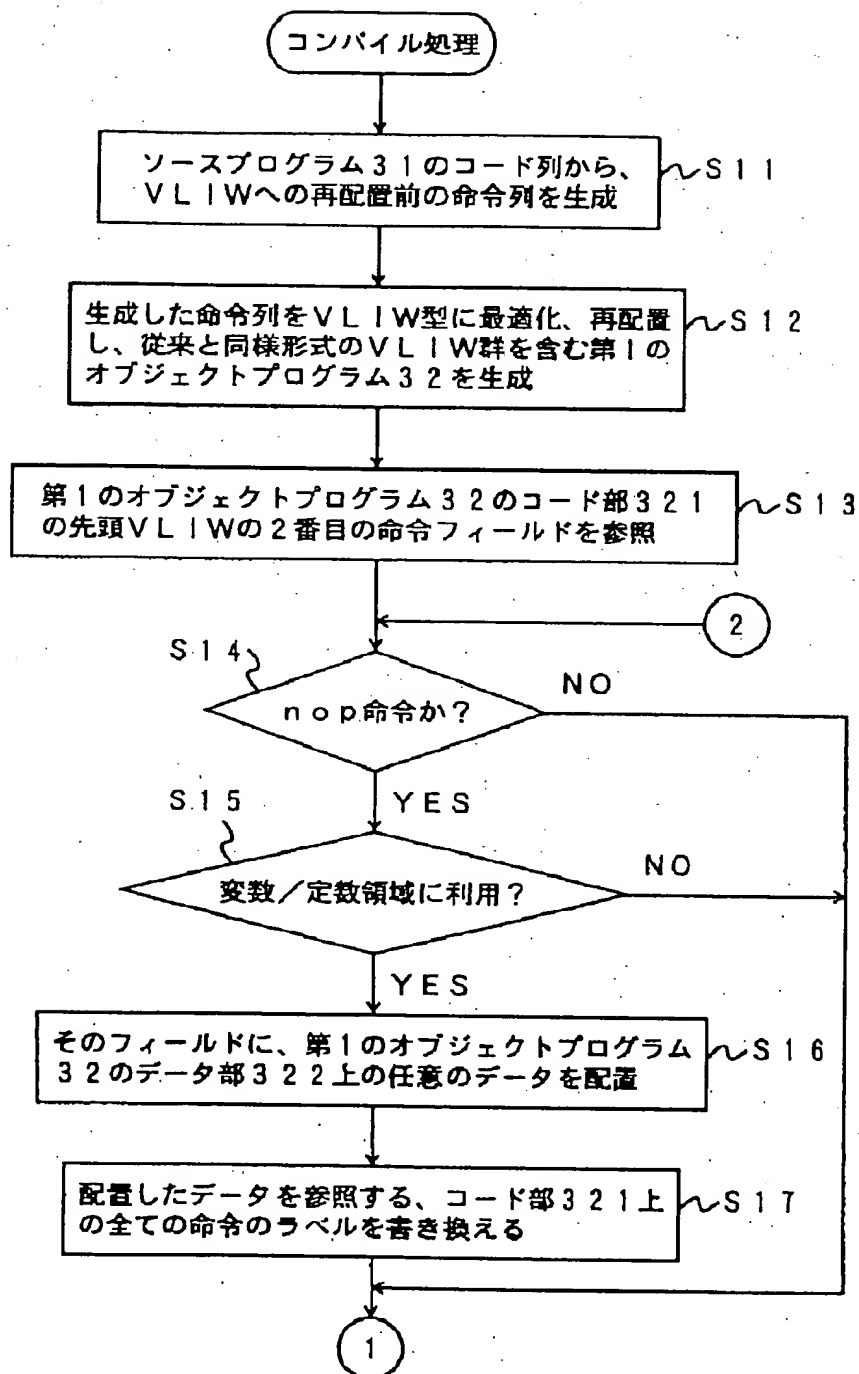




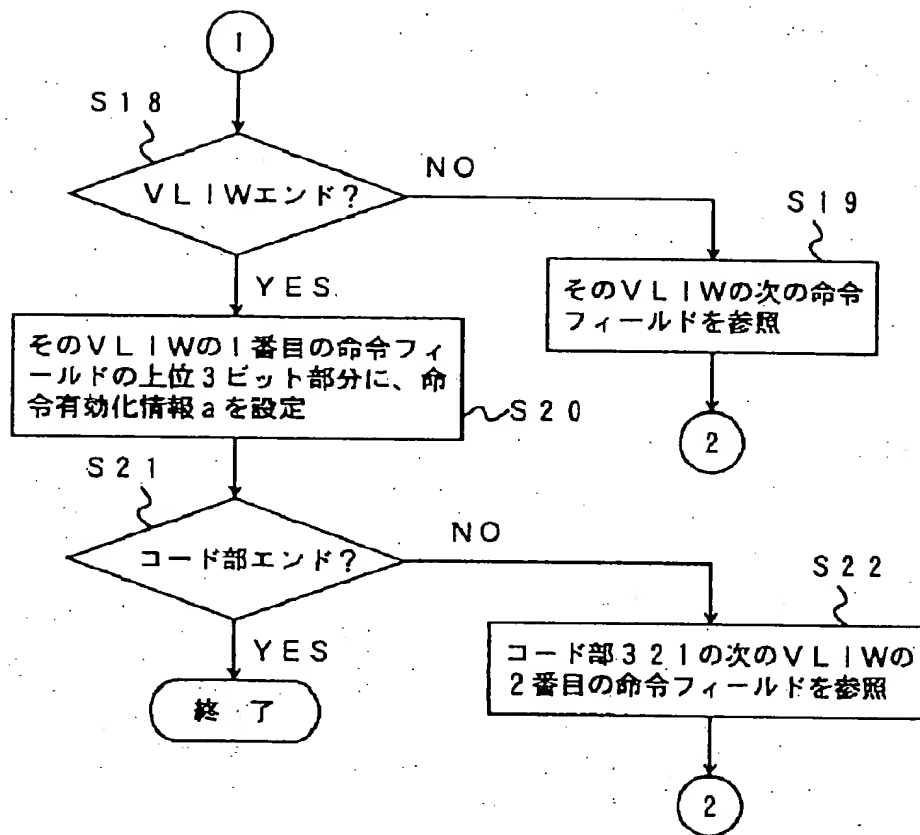
【図4】



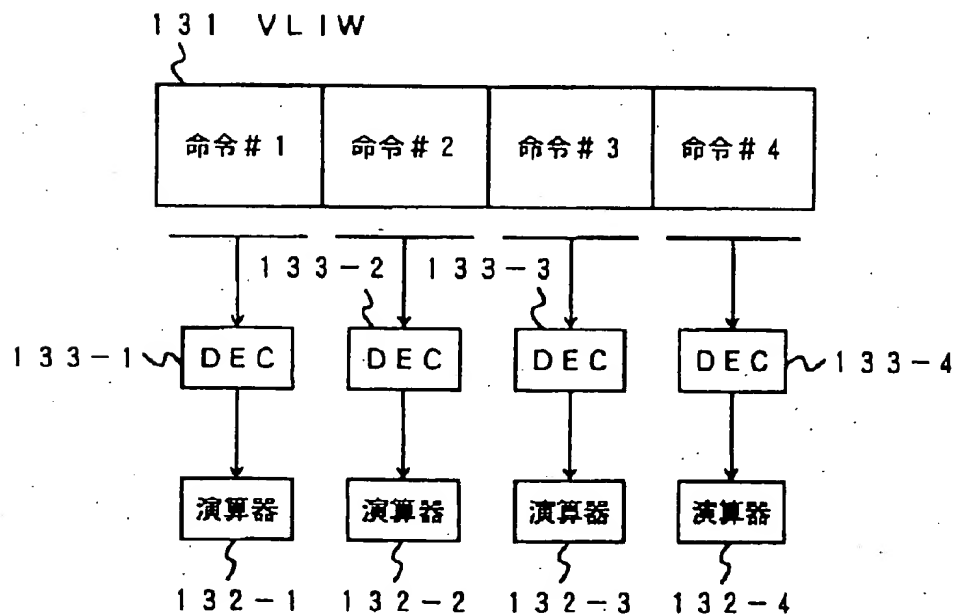
【図 5】



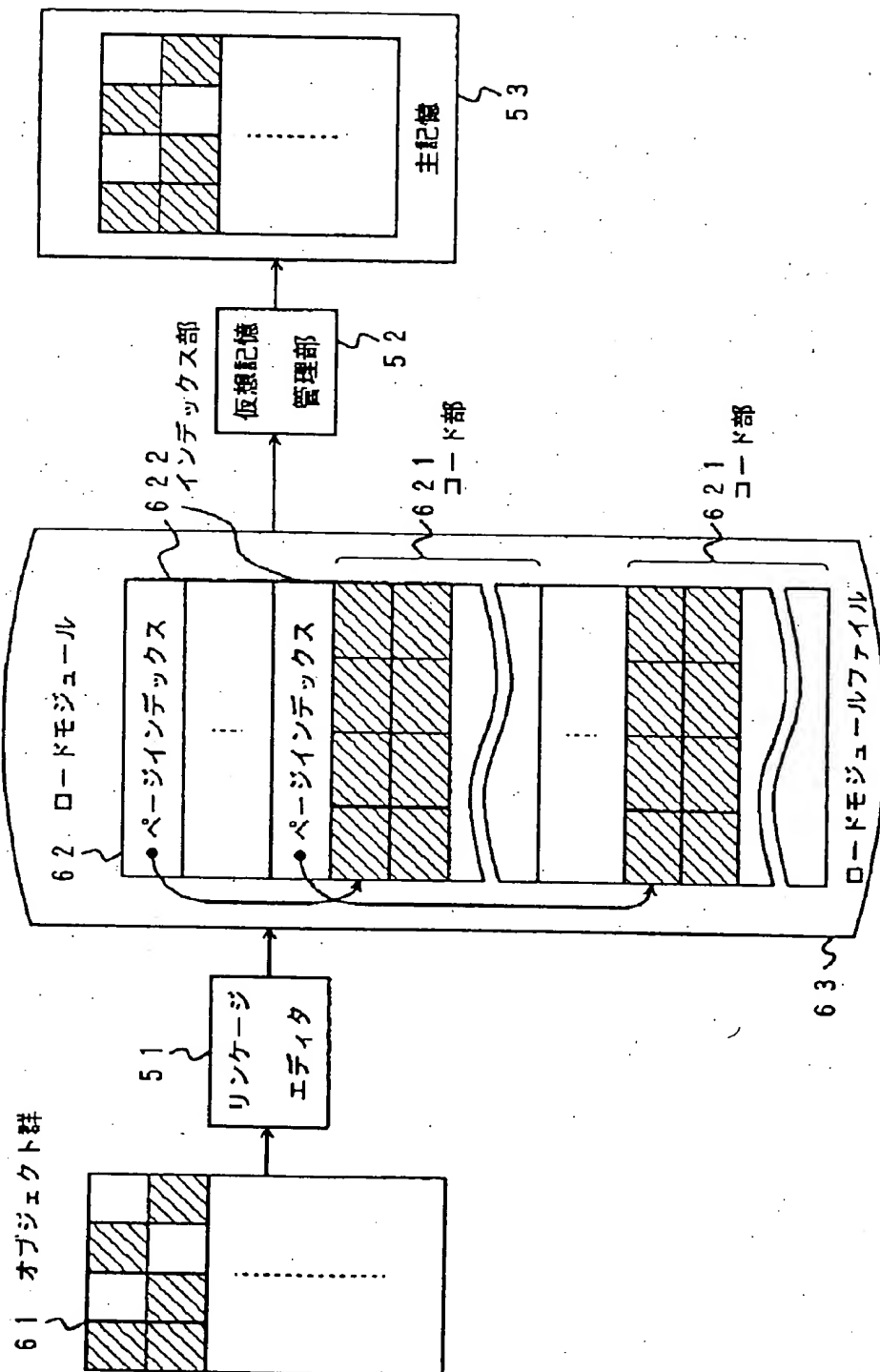
【図6】



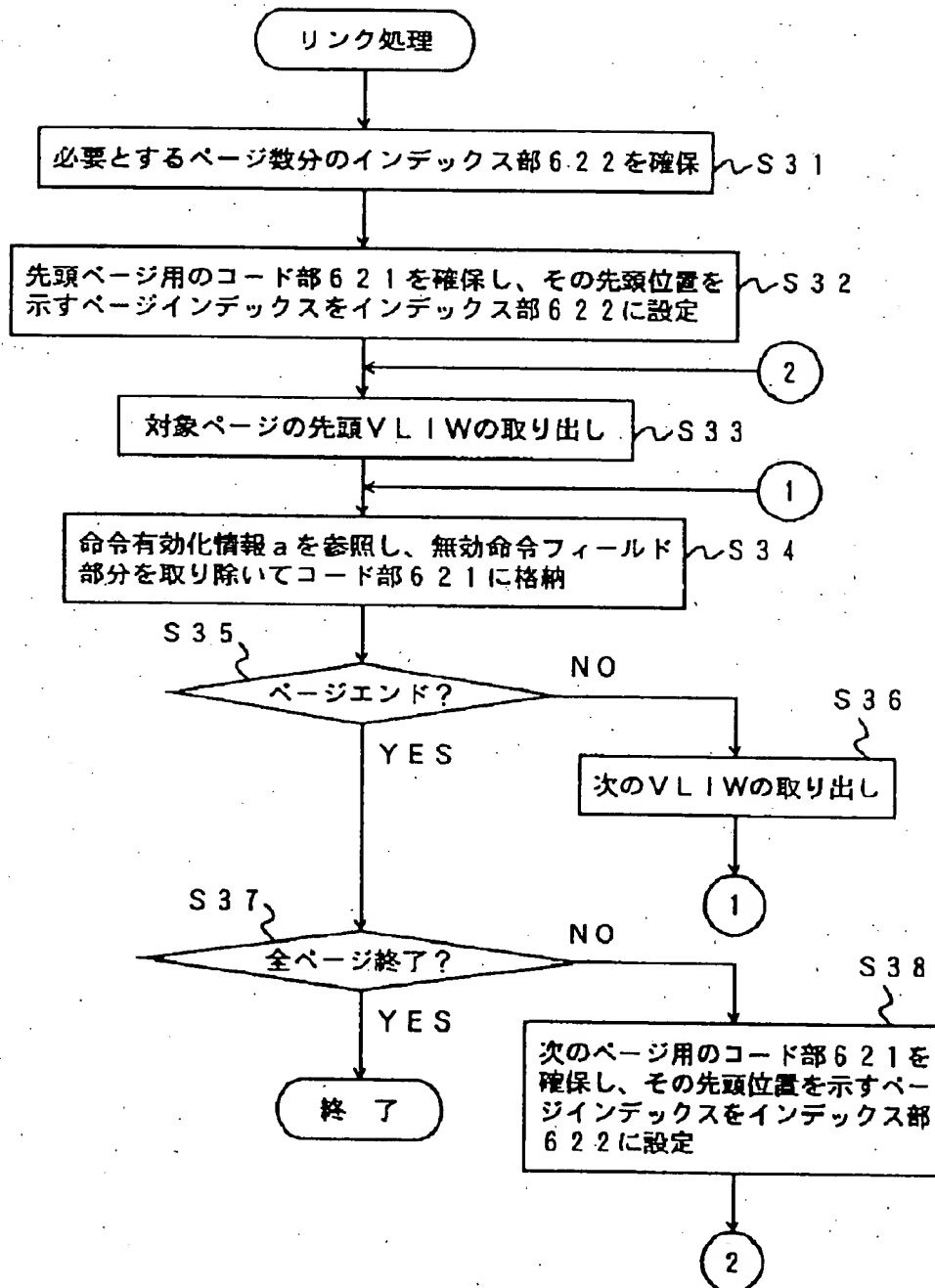
【図13】



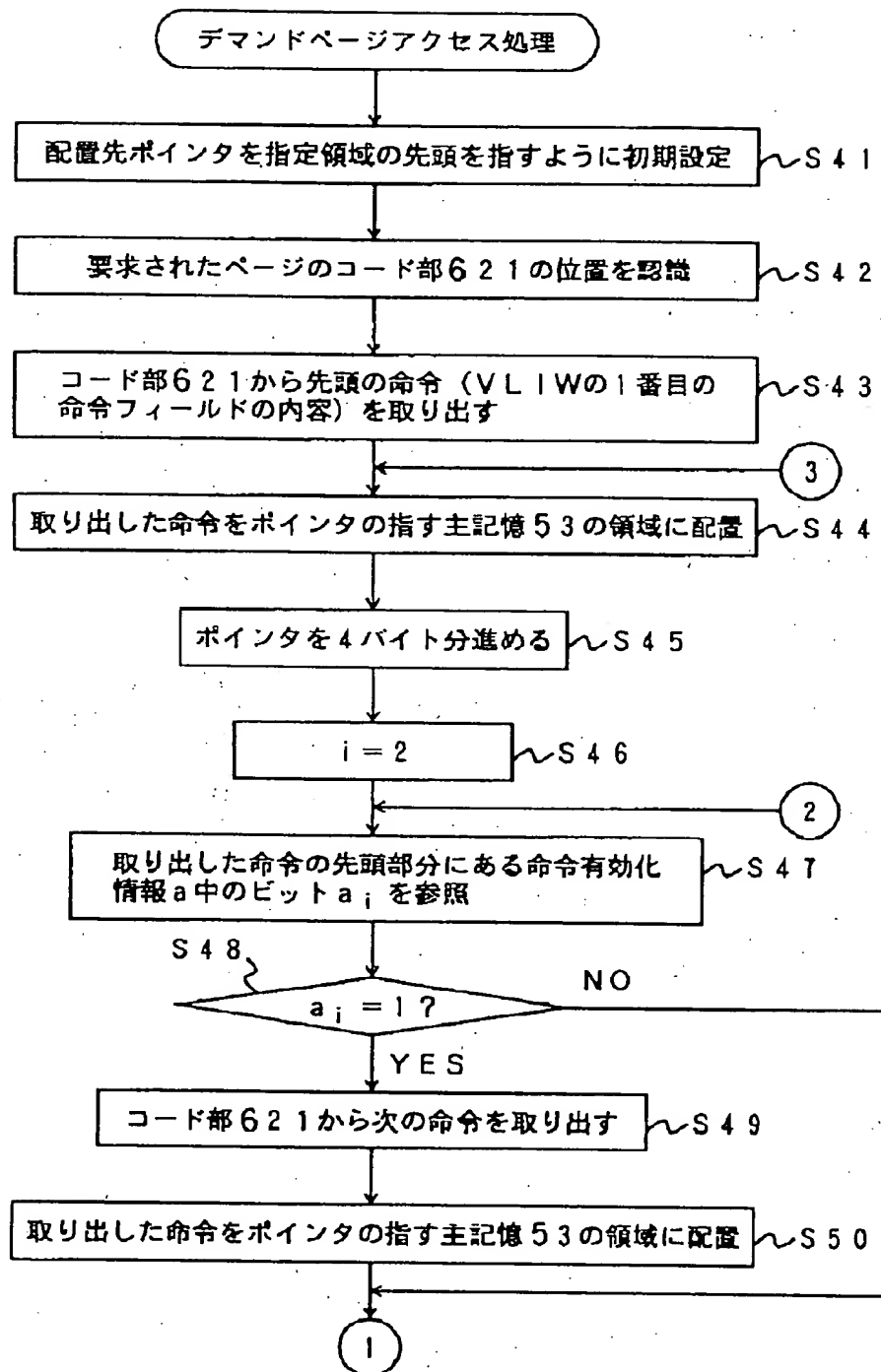
【図7】



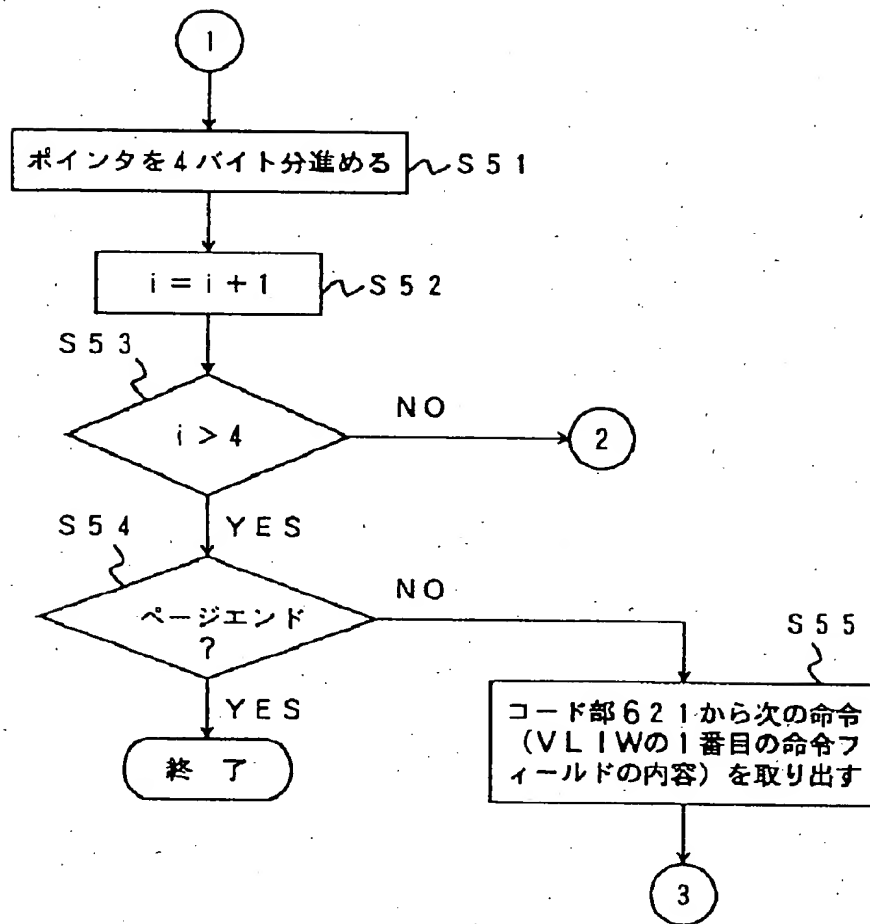
【図 8】



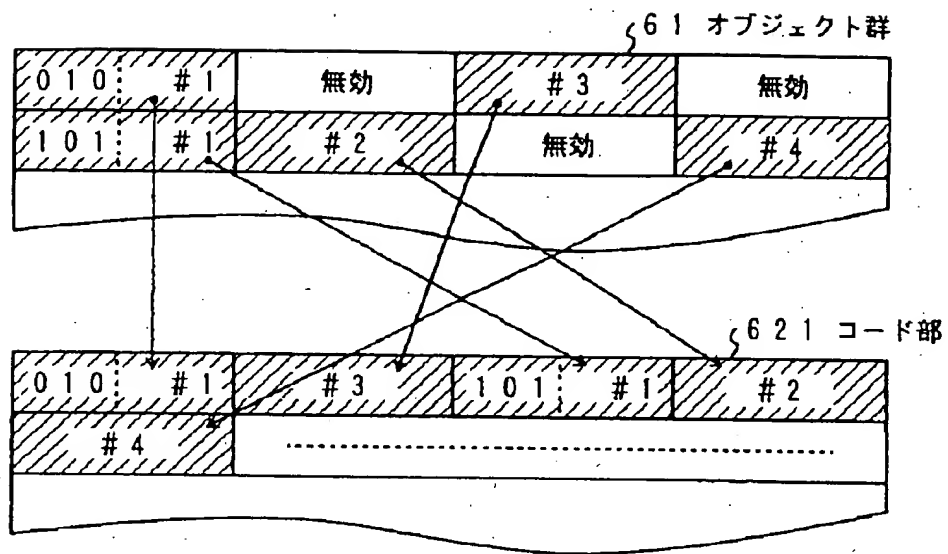
【図 9】



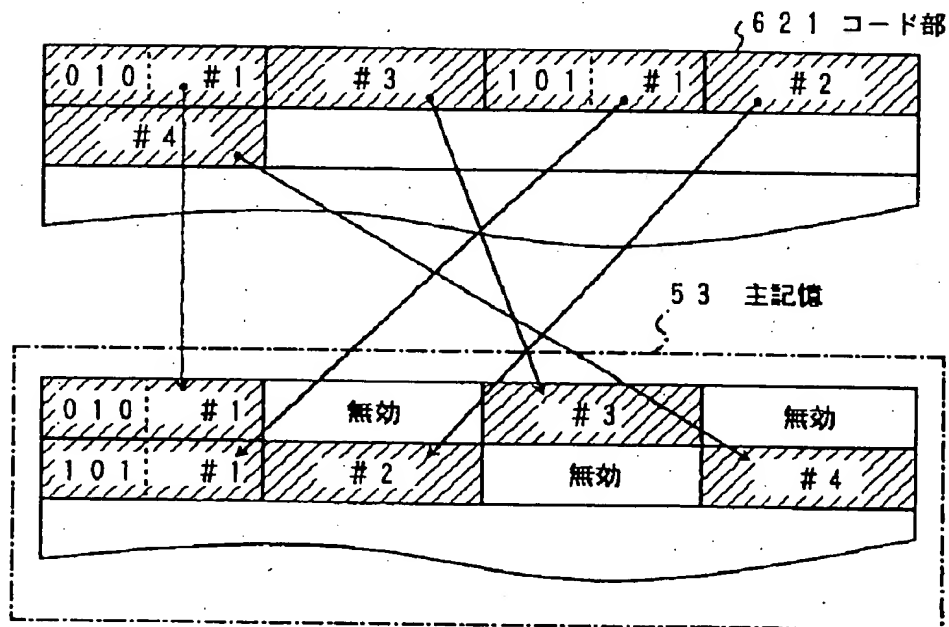
【図10】



【図 1.1】



(a)



(b)



【図 12】

